Ejercicios

Unidad 12

Ficheros (B)

EJERCICIOS (B) – LECTURA Y ESCRITURA DE FICHEROS

Para probar estos ejercicios utilizar el archivo “Documentos.zip”.

Ejercicio B1 - Máximo y mínimo

Implementa un programa que muestre por pantalla los valores máximos y mínimos del archivo ‘numeros.txt’.

Ejercicio B2 - Notas de alumnos

El archivo ‘alumnos_notas.txt’ contiene una lista de 10 alumnos y las notas que han obtenido en cada asignatura. El número de asignaturas de cada alumno es variable. Implementa un programa que muestre por pantalla la nota media de cada alumno junto a su nombre y apellido, ordenado por nota media de mayor a menor.

Ejercicio B3 - Ordenando archivos

Implementa un programa que pida al usuario un nombre de archivo A para lectura y otro nombre de archivo B para escritura. Leerá el contenido del archivo A (por ejemplo ‘usa_personas.txt’) y lo escribirá ordenado alfabéticamente en B (por ejemplo ‘usa_personas_sorted.txt’).

Ejercicio B4 - Nombre y apellidos

Implementa un programa que genere aleatoriamente nombres de persona (combinando nombres y apellidos de ‘usa_nombres.txt’ y ‘usa_apellidos.txt’). Se le pedirá al usuario cuántos nombres de persona desea generar y a qué archivo añadirlos (por ejemplo ‘usa_personas.txt’).

Ejercicio B5 - Diccionario

Implementa un programa que cree la carpeta ‘Diccionario’ con tantos archivos como letras del abecedario (A.txt, B.txt… Z.txt). Introducirá en cada archivo las palabras de ‘diccionario.txt’ que comiencen por dicha letra.

Ejercicio B6 - Búsqueda en PI

Implementa un programa que pida al usuario un número de cualquier longitud, como por ejemplo “1234”, y le diga al usuario si dicho número aparece en el primer millón de

decimales del nº pi (están en el archivo 'pi-million.txt'). No está permitido utilizar ninguna librería ni clase ni método que realice la búsqueda. Debes implementar el algoritmo de búsqueda tú.

Ejercicio B7 - Estadísticas

Implementa un programa que lea un documento de texto y muestre por pantalla algunos datos estadísticos: nº de líneas, nº de palabras, nº de caracteres y cuáles son las 10 palabras más comunes (y cuántas veces aparecen). Prueba el programa con los archivos de la carpeta 'Libros'.

NOTA: *Para llevar la cuenta de cuántas veces aparece cada palabra puedes utilizar una HashTable. Una tabla hash es una estructura de datos tipo colección (como el ArrayList), que permite almacenar pares clave-valor. Por ejemplo {"elefante", 5} o {"casa", 10} son pares <String,Integer> que asocian una palabra (clave) con un nº entero (valor).*

Ejercicio B8 – Jugando con csv

Copia el fichero Persona.jar en el directorio de tu proyecto. Añadirlo como librería. Ya podemos usar la clase utilidades.Persona.

En el fichero datos_personas.csv del directorio Documentos tenemos datos de personas. Crea un mapa para guardar todos los datos de las personas, referenciadas por su dni.

Lee el fichero, crea objetos Persona con sus datos y añádelos al mapa.

A continuación pregunta por un dni, y mostrará los datos de esa persona si se encuentra en el sistema, o informará que no se encuentra. Se repetirá hasta que se introduzca "fin".

Para finalizar genera un fichero en el directorio Documentos, con nombre datos_junilados.csv con los datos en formato "dni;nombre;apellidos;edad" de las personas jubiladas, ordenadas por dni.

Ejercicio B9 - Serialización

En el fichero datos_personas.bin se encuentra serializada una lista de personas. Recupera los datos, crea otra lista con aquellas personas con edad entre 20 y 29 años y guardala serializada en el fichero datos_veintes.ser

Ejercicio B10 - Uso de ficheros properties

Supongamos que necesitas desarrollar una aplicación que permita a los usuarios configurar ciertos parámetros, como por ejemplo el nombre de la empresa, el número máximo de intentos de inicio de sesión, el tiempo de expiración de sesión, etc.

Para ello, podrías utilizar un archivo de properties donde se almacenen estas configuraciones y que la aplicación pueda leer en tiempo de ejecución.

El ejercicio consiste en:

1. Crear un archivo `config.properties` con las siguientes propiedades y sus respectivos valores:

```
empresa.nombre=Mi empresa
login.max_intentos=3
sesion.timeout=1800
```

2. Crear una clase `ConfigManager` que tenga un método `loadConfig()` que lea el archivo de properties y cargue sus valores en un objeto `Properties`. Este método deberá lanzar una excepción si el archivo no existe o no se puede leer.
3. Agregar los siguientes métodos a la clase `ConfigManager`:

- `getNombreEmpresa()`: Devuelve el nombre de la empresa.
- `getMaxIntentos()`: Devuelve el número máximo de intentos de inicio de sesión. Es un entero
- `getSesionTimeout()`: Devuelve el tiempo de expiración de sesión en segundos. Es un entero

4. Crear una clase `EjercicioB10` que utilice la clase `ConfigManager` para imprimir por pantalla los valores de configuración.

Ejercicio B11 - Uso de JSON

En este ejercicio, vamos a crear una pequeña aplicación que permita almacenar información sobre una lista de tareas en un archivo JSON y leer la información de ese archivo para mostrarla en pantalla. Todas las clases pueden estar en el mismo fichero, solo puede ser pública la que contiene el método main y necesitarás añadir la librería gson o jackson a tu proyecto.

1. Crea una clase `Task` con los siguientes atributos:

- `id` (int): identificador único de la tarea.
- `description` (String): descripción de la tarea.
- `priority` (int): prioridad de la tarea (1 para alta, 2 para media, 3 para baja).
- `completed` (boolean): indica si la tarea ha sido completada o no.

2. Crea una clase `TaskManager` que tenga los siguientes métodos:

- `addTask(Task task)`: añade una tarea a la lista de tareas.
- `removeTask(int id)`: elimina la tarea con el identificador especificado de la lista de tareas.

- `getTask(int id)`: devuelve la tarea con el identificador especificado.
- `getTasks()`: devuelve todas las tareas en forma de lista.
- `saveTasksToFile(String fileName)`: guarda la lista de tareas en un archivo JSON con el nombre especificado.
- `loadTasksFromFile(String fileName)`: carga la lista de tareas desde un archivo JSON con el nombre especificado.

3. Implementa los métodos de la clase `TaskManager` para que manipulen la lista de tareas y permitan guardar y cargar la información desde un archivo JSON.
4. Crea una clase `EjercicioB11` que tenga un método `main` que haga lo siguiente:

- Crea un objeto `TaskManager`.
- Añade varias tareas a la lista de tareas del `TaskManager`.
- Guarda la lista de tareas en un archivo JSON.
- Carga la lista de tareas desde el archivo JSON.
- Muestra la lista de tareas en pantalla. (¿Qué necesitamos?)

5. Ejecuta la aplicación y comprueba que se han guardado y cargado correctamente las tareas desde el archivo JSON.